

11 0 00-00

71

11/03/00

1c957 U.S. PTO

**UTILITY PATENT APPLICATION TRANSMITTAL  
(Large Entity)***(Only for new nonprovisional applications under 37 CFR 1.53(b))*Docket No.  
LUC-731USTotal Pages in this Submission  
40**TO THE ASSISTANT COMMISSIONER FOR PATENTS**Box Patent Application  
Washington, D.C. 20231

Transmitted herewith for filing under 35 U.S.C. 111(a) and 37 C.F.R. 1.53(b) is a new utility patent application for an invention entitled:

**COMMUNICATION PROTOCOL FOR DATA EXCHANGE VIA SHARED FILES**1c922 U.S. PTO  
09/705578

11/03/00

and invented by:

Valentin Panayotov

If a **CONTINUATION APPLICATION**, check appropriate box and supply the requisite information:☐ Continuation ☐ Divisional ☐ Continuation-in-part (CIP) of prior application No.: \_\_\_\_\_

Which is a:

☐ Continuation ☐ Divisional ☐ Continuation-in-part (CIP) of prior application No.: \_\_\_\_\_

Which is a:

☐ Continuation ☐ Divisional ☐ Continuation-in-part (CIP) of prior application No.: \_\_\_\_\_

Enclosed are:

**Application Elements**

1. ☒ Filing fee as calculated and transmitted as described below
2. ☒ Specification having 22 pages and including the following:
  - a. ☒ Descriptive Title of the Invention
  - b. ☐ Cross References to Related Applications *(if applicable)*
  - c. ☐ Statement Regarding Federally-sponsored Research/Development *(if applicable)*
  - d. ☐ Reference to Microfiche Appendix *(if applicable)*
  - e. ☒ Background of the Invention
  - f. ☒ Brief Summary of the Invention
  - g. ☒ Brief Description of the Drawings *(if drawings filed)*
  - h. ☒ Detailed Description
  - i. ☒ Claim(s) as Classified Below
  - j. ☒ Abstract of the Disclosure

# UTILITY PATENT APPLICATION TRANSMITTAL (Large Entity)

(Only for new nonprovisional applications under 37 CFR 1.53(b))

Docket No.  
LUC-731US

Total Pages in this Submission  
40

## Application Elements (Continued)

3. ☒ Drawing(s) (when necessary as prescribed by 35 USC 113)
- a. ☐ Formal Number of Sheets \_\_\_\_\_
- b. ☒ Informal Number of Sheets 1
4. ☒ Oath or Declaration
- a. ☒ Newly executed (original or copy) ☐ Unexecuted
- b. ☐ Copy from a prior application (37 CFR 1.63(d)) (for continuation/divisional application only)
- c. ☒ With Power of Attorney ☐ Without Power of Attorney
- d. ☐ DELETION OF INVENTOR(S)  
Signed statement attached deleting inventor(s) named in the prior application,  
see 37 C.F.R. 1.63(d)(2) and 1.33(b).
5. ☐ Incorporation By Reference (usable if Box 4b is checked)  
The entire disclosure of the prior application, from which a copy of the oath or declaration is supplied under Box 4b, is considered as being part of the disclosure of the accompanying application and is hereby incorporated by reference therein.
6. ☐ Computer Program in Microfiche (Appendix)
7. ☐ Nucleotide and/or Amino Acid Sequence Submission (if applicable, all must be included)
- a. ☐ Paper Copy
- b. ☐ Computer Readable Copy (identical to computer copy)
- c. ☐ Statement Verifying Identical Paper and Computer Readable Copy

## Accompanying Application Parts

8. ☒ Assignment Papers (cover sheet & document(s))
9. ☐ 37 CFR 3.73(B) Statement (when there is an assignee)
10. ☐ English Translation Document (if applicable)
11. ☒ Information Disclosure Statement/PTO-1449 ☒ Copies of IDS Citations
12. ☐ Preliminary Amendment
13. ☒ Acknowledgment postcard
14. ☒ Certificate of Mailing
- ☐ First Class ☒ Express Mail (Specify Label No.): EL684159475US

**UTILITY PATENT APPLICATION TRANSMITTAL**  
**(Large Entity)**

*(Only for new nonprovisional applications under 37 CFR 1.53(b))*

Docket No.  
LUC-731US

Total Pages in this Submission  
40

**Accompanying Application Parts (Continued)**

15. ☐ Certified Copy of Priority Document(s) *(if foreign priority is claimed)*

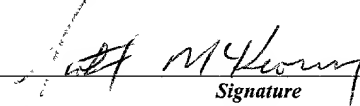
16. ☐ Additional Enclosures *(please identify below):*

**Fee Calculation and Transmittal**

**CLAIMS AS FILED**

For	#Filed	#Allowed	#Extra	Rate	Fee
Total Claims	10	- 20 =	0	x \$18.00	\$0.00
Indep. Claims	4	- 3 =	1	x \$80.00	\$80.00
Multiple Dependent Claims (check if applicable) <input type="checkbox"/>					\$0.00
BASIC FEE					\$690.00
OTHER FEE (specify purpose) <u>Assignment Recordation</u>					\$40.00
TOTAL FILING FEE					\$810.00

- ☐ A check in the amount of \_\_\_\_\_ to cover the filing fee is enclosed.
- ☒ The Commissioner is hereby authorized to charge and credit Deposit Account No. **12-2325** as described below. A duplicate copy of this sheet is enclosed.
- ☒ Charge the amount of **\$810.00** as filing fee.
  - ☒ Credit any overpayment.
  - ☒ Charge any additional filing fees required under 37 C.F.R. 1.16 and 1.17.
  - ☐ Charge the issue fee set in 37 C.F.R. 1.18 at the mailing of the Notice of Allowance, pursuant to 37 C.F.R. 1.311(b).

  
Signature

Allan Ratner, Reg. No. 19,717  
Scott A. McKeown, Reg. No. 42,866  
Attorneys for Applicant  
RATNER & PRESTIA  
Suite 301, One Westlakes (Berwyn)  
P.O. Box 980  
Valley Forge, PA 19482-0980  
(610) 407-0700

Dated: November 3, 2000

cc:

**CERTIFICATE OF MAILING BY "EXPRESS MAIL" (37 CFR 1.10)**

Applicant(s):

Docket No.

LUC-731US

Serial No.

Not Yet Assigned

Filing Date

Herewith

Examiner

Group Art Unit

Invention: **COMMUNICATION PROTOCOL FOR DATA EXCHANGE VIA SHARED FILES**Jc922 U.S. PTO  
09/705578

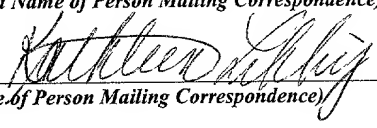
11/03/00

I hereby certify that the following correspondence:

Utility Patent Application and accompanying documents

*(Identify type of correspondence)*

is being deposited with the United States Postal Service "Express Mail Post Office to Addressee" service under  
37 CFR 1.10 in an envelope addressed to: The Assistant Commissioner for Patents, Washington, D.C. 20231 on

November 3, 2000*(Date)*Kathleen Libby*(Typed or Printed Name of Person Mailing Correspondence)*  
*(Signature of Person Mailing Correspondence)*EL684159475US*("Express Mail" Mailing Label Number)***Note: Each paper must have its own certificate of mailing.**

## COMMUNICATION PROTOCOL FOR DATA EXCHANGE VIA SHARED FILES

## BACKGROUND OF THE INVENTION

5 The invention relates to data exchange between computer applications. More particularly, the invention relates to the exchange of data between application level shared files to establish a communication protocol between two computer applications.

10 Computer technology has fostered the development of computer software applications which have become integral components of commercial as well as personal computing systems. As such, the need for computers supporting common applications to exchange data has resulted in a proliferation of network technologies and communication standards and protocols. These technologies typically address the management of dissimilar physical, link and network layers of the OSI (Open Systems Interconnection) model, for the purpose of establishing a viable communication medium between applications and computing devices.

15 For example, data exchange between computer applications may include, (i) computer control and data acquisition of automated equipment data, followed by (ii) processing of the acquired data, and finally by (iii) archiving/export to a database. Presently, any of these functions can be performed by individual computer applications. However, for continuous operation, the separate computer applications must be simultaneously running (preferably on a single machine) and freely  
20 exchanging data with each other. Since real-time communication between two computer applications is often impractical (except for the case when they happen to be from the same generation of the same manufacturer) more often than not, a single computer application has to be developed having all of the above functionality; This is a very costly, time consuming, and an entirely non-portable solution.

Some industry organizations have established standards and protocols to further enhance data communication capabilities. For example, the Semiconductor Equipment and Materials International (SEMI) trade association has introduced standards such as SEMI E4 (SEMI Equipment Communications Standard 1 Message Transfer -- SECS-I), SEMI E5 (SEMI Equipment Communications Standard 2 Message Content -- SECS-II), SEMI E30 (Generic Model for Communications and Control of Manufacturing Equipment -- GEM), SEMI E37 (High-Speed SECS Message Services Generic Services -- HSMS). These standards specify strict message (data) formats for communication over a dedicated hardware line (RS 232 serial connection or a TCP/IP stream support). However, the standards (i) pertain only to the particular case of data exchange between a piece of semiconductor equipment and a host computer, (ii) require additional hardware connectivity, and (iii) require very expensive, often proprietary, and generally non-portable software and hardware development.

Accordingly, there is a need for a simple communication protocol in which a broad range of computer applications can communicate with each other.

### SUMMARY OF THE INVENTION

The present invention is a system and associated method of exchanging data between a first and second computer application of a computer system. A computer application data file receives data from the first computer application. A computer application send file receives notification when the computer application data file has received data from the first computer application. A computer application read file receives notification when data has been read from the computer application data file by the second computer application. The first computer application monitors the computer application read file for notification from the second computer application to initiate further writing to the computer application data file.

According to one aspect of the invention, a computing system for exchanging data between computer applications of the system is provided. A computer application data file for each computer application receives data from a corresponding one of the computer applications. A computer application send file corresponding to

each computer application data file receives notification that the corresponding computer application data file has received data from the corresponding one of the computer applications. A computer application read file corresponding to each computer application data file receives notification that data has been read from the computer application data file by a non-corresponding computer application. The corresponding computer application monitors the computer application read file for notification to initiate further writing to the corresponding computer application data file.

According to another aspect of the invention, data originating from the first computer application is written to a first computer application data file. A first computer application send file is notified when data has been written to the first computer application data file by the first computer application. The first computer application send file is monitored from the second computer application for notification that data has been written to the first computer application data file by the first computer application. The data of the first computer application data file is read from the second computer application upon notification. A first computer application read file is notified when data has been read by the second computer application from the first computer application data file.

According to still another aspect of the invention, a system and associated method of exchanging bi-directional data between a first and second computer application of a computer system is provided. Data originating from the second computer application is written to a second computer application data file. A second computer application send file is notified when data has been written to the second computer application data file by the second computer application. The second computer application send file is monitored from the first computer application for notification that data has been written to the second computer application data file by the second computer application. The data of the second computer application data file is read from the first computer application upon notification. The second computer application read file is notified when data has been read by the first computer application from the second computer application data file.

## BRIEF DESCRIPTION OF THE SEVERAL VIEWS OF THE DRAWINGS

The foregoing summary as well as the following detailed description of exemplary embodiments of the invention, will be better understood when read in conjunction with the appended drawing. For the purpose of illustrating the invention, there is shown in the drawing an exemplary embodiment of the invention. It should be understood, however, that the invention is not limited to the precise arrangement and instrumentality shown. In the drawing:

Fig. 1 is a block diagram of an embodiment of a preferred communication flow architecture of the present invention.

## DETAILED DESCRIPTION OF THE INVENTION

Certain terminology is used herein for convenience only and is not to be taken as a limitation on the present invention. The term "computer application" is defined as any a set of instructions, specific for a given microprocessor, that when executed by that microprocessor will specify a valid sequence of predefined operations. The term "communication protocol" is defined as a sequence of states and events that specify the procedure of exchange of information at the application level between two computer applications. The term "communication link" is defined as the one-directional connection between two computer applications where one application is the source of the data and the other the destination for the data.

A portion of this patent document contains material which is subject to copyright protection. The owner has no objection to the facsimile reproduction by anyone of the patent document or patent disclosure as it appears in the Patent and Trademark Office patent file records, but otherwise reserves all rights whatsoever.

The present invention concerns a system and associated method for exchanging data between computer applications via shared file structures and a predefined flow sequence. The applications utilize basic input/output functionality. For each communication link, one data file and two notification files are used.

A computer application data file receives data from the first computer application. A computer application send file receives notification when the computer



application data file has received data from the first computer application. A computer application read file receives notification when data has been read from the computer application data file by the second computer application. The first computer application monitors the computer application read file for notification from the second computer application to initiate further writing to the computer application data file.

The communication protocol described herein provides the necessary mechanism for synchronization and data validation. In addition, it inherently guards against sharing violations, since each computer application can simultaneously write or "talk" and read or "listen", this is of utmost importance. As defined by most computer operating systems, a sharing violation is an attempt by two or more computer applications to simultaneously access a given file in a write mode or to operate on an otherwise protected file. A straightforward attempt for exchange of data between two computer applications via shared files will fail because of sharing violations. The present communication protocol allows only one of the applications to access a given file in a write mode. Monitoring of the notification files and following the indicated data flow sequence prevents access attempts to files that are currently used by the other application. An application level handshaking protocol is thus established that guarantees against sharing violations.

Referring now to Fig. 1, two computer applications A and B are shown for establishing a two-way or bi-directional communication. A set of three shared files in the familiar UNIX/DOS notation (i.e., A\_data.\*) are designated for transfer of data from computer application A to computer application B (top oblong circle). Another set of three files (B\_data.\*) are designated for transfer of data from application B to application A (bottom oblong circle). The computing system supporting applications A and B, and communicating with the data and communication files can be a single or "stand-alone" computing device (in a multitasking mode) or separate computing devices connected via an existing network (not shown). It is recognized by those skilled in the art that the present invention is not limited to UNIX/DOS operating systems and/or a particular computer application data format.

In general, for every additional communication link (i.e., oblong circle of Fig. 1) another set of notification and data files are used. For a meaningful

communication to occur, the data exchange needs to be synchronized by the notification files. For example, the data exchange will fail if computer application B tries to read data before it is written by computer application A. The need for synchronization is a necessary requirement in the field of communications and communication protocols must specify the correct sequence of the data exchange events.

The data file designated "A\_data.dat" in Fig. 1 will contain the actual computer application data being provided from computer application A for exchange to computer application B. The data may be any valid digital content. A will only write to the A\_data.dat file and B will only read from the A\_data.dat file.

The send notification file designated "A\_data.snt" in Fig. 1 will contain a predefined pair of values (e.g., 1 and 0). The value will indicate whether valid data is present (1) or absent (0) in the A\_data.dat file (i.e., whether or not application A has written data). Computer application A will only write to the send notification file A\_data.snt and computer application B will only read from the send notification file A\_data.snt.

The read notification file designated "A\_data.red" in Fig. 1 will contain a predefined pair of values (e.g., 1 or 0). The value will indicate whether data has been successfully read (1) or not (0) from the data file. Computer application A will only read from the read notification file A\_data.red and computer application B will only write to the read notification file A\_data.red.

Using the above set of communication files, the following minimum set of events can define a valid data transfer of data from application A to application B:

1. Computer application A "writes" data to the data file A\_data.dat.
2. Computer application A notifies computer application B that data is present in the data file by writing a value to file A\_data.snt.
3. Computer application B "reads" data from the data file A\_data.dat.

4. Computer application B notifies computer application A through the notification file that the data has been read from the data file by writing a value to file A\_data.red.

The above sequence may need to be refined and expanded if the particular operating system that hosts the data and notification files places additional restrictions on sharing and timing access.

The following detailed list of the states (of both applications and communication files) and events defines the communication protocol for data transfer from application A to application B. This is a single communication link utilizing three shared files (top oblong circle on Fig. 1). Bi-directional data exchange would be the simultaneous operation of (2) two such data links and six shared files (top and bottom oblong circles in Fig. 1)

#### Initialization

At start up, before any data transfer is initiated, the initial/stand-by states of the communication files should be set to the following:

- Send Notification File                      contains "clear/invalid data"
- Read Notification File                      contains "clear/unsuccessfully read"
- Data File                                      irrelevant

The initial states of the computer applications should be set to the following:

- Application A                                  Any state -- ready to initiate the transfer.
- Application B                                  Idle state -- waits for a "valid data present" indication in the Send Notification File.

The proper initialization procedure requires that application A write a "clear/invalid data" notification to the send notification file to ensure that any data that may be left over from prior communications is overwritten (this action will prevent B from reading any data that may be left over from prior communications in the Data File). Application B needs to write an "clear/unsuccessfully read" to the read

notification file to ensure that any data that may be left over from prior communications be overwritten. The contents of the data file are irrelevant, previous data will not be inadvertently read and would be overwritten by the first data transfer.

### Data Transfer

- 5 For a single data transfer in this communication link the following sequence is defined:

**TABLE 1**

	<b>Application A</b>	<b>Application B</b>
1.	Writes data to the A_dat.dat file.	Idle state – waits for a "valid data present" indication in the A_data.snt send notification file.
2.	Writes a "valid data present" indication to the A_data.snt send notification file. Enters idle state.	Idle state -- waits for a "valid data present" indication in the A_data.snt send notification file.
3.	Idle state – waits for a "successfully read" indication in the A_data.red read notification file.	Reads the "valid data present" indication from the A_data.snt send notification file. Exits idle state.
4.	Idle state – waits for a "successfully read" indication in the A_data.red read notification file.	Reads the data from the A_data.dat data file.
5.	Idle state – waits for a "successfully read" indication in the A_data.red read notification file.	Writes a "successfully read" indication to the A_data.red read notification file. Enters idle state.
6.	Reads the "successfully read" indication from the A_data.red read notification file. Exits idle state.	Idle state – waits for a "clear/invalid data" indication in the Send Notification File.
7.	Writes a "clear/invalid data" indication to the A_data.snt send notification file. Enters idle state.	Idle state – waits for a "clear/invalid data" indication in the A_data.snt send notification file.
8.	Idle state – waits for a "clear/unsuccessfully read" indication in the A_data.red read notification file.	Reads the "clear/invalid data" indication from the A_data.snt send notification file. Exits idle state.
9.	Idle state – waits for a "clear/unsuccessfully read" indication in the A_data.red read notification	Writes a "clear/unsuccessfully read" indication to the A_data.red read notification file. Enters idle state.

	Application A	Application B
	file.	
10.	Reads the "clear/unsuccessfully read" indication from the A_data.red read notification file. Data transfer is over. Ready to initiate next transfer.	Idle state – waits for a "valid data present" indication in the A_data.snt send notification file. This is the stand-by state – transfer is over, waiting for next transfer.

At the end of this sequence all files are in their initial/stand-by state, application B is in a idle mode expecting application A to initiate the next data transfer.

The present invention enables the interface of computer applications running on dissimilar or remote computer systems connected over an existing network; interface of existing computer controlled equipment without the need for additional hardware connectivity; remote control of computer applications; and interface of computer applications from different software generations.

Commented source code of an embodiment of the communication protocol for data exchange via shared files follows below.

#### Visual Basic Implementation

The following is commented MICROSOFT® Visual Basic source code for implementing the functionality of the present invention. The commented sections in the code describe the calling sequences in detail. The implementation provides (i) automatic creation and consistent naming of all notification files in order to safeguard against inadvertent mis-configurations, (ii) an externally accessible variable for termination in case of communication error, and (iii) standard Visual Basic error trapping procedures, execution yielding, etc.

```

'-----
' CP_Util.bas      Version 3.0
'
' Visual Basic V5.0 implementation of Communication Protocol for Data
' Exchange Via Shared Files
'
' For each data link, an application has to declare a file path and a
' communication link number. The application can be either a "data

```

' source" or a "data destination" in that link.

' For example, an application that will be a "data source"  
' in a given link may be initialized with the following:

```
' sPath = "C:\WINDOWS\TEMP\"
' sLink = 712
' Call CP_InitSend(sPath, sLink, sLinkFile, sDataFile)
```

' while the corresponding "data destination" application will be  
' initialized with:

```
' dPath = "C:\WINDOWS\TEMP\"
' dLink = 712
' Call CP_InitReceive(dPath, dLink, dLinkFile, dDataFile)
```

' Then for every transfer the "data source" application needs to  
' execute the following code:

```
' Call WriteData(sDataFile,Data)
' Call CP_SendData(sLinkFile)
```

' The "data destination" application receives the data by executing:

```
' Call CP_WaitForData(dLinkFile)
' Call ReadData(dDataFile,Data)
' Call CP_ReceiveData(dLinkFile)
```

' WriteData and ReadData may be any custom subroutines designed to  
' write/read a compatible data format to/from the data file.

#### Option Explicit

'----- public variable providing for externally forced exit  
Public CP\_Abort As Boolean

'----- private constants defining communication role

Private Const CP\_Source As String = "0"

Private Const CP\_Destination As String = "1"

```
'----- private constants defining flag contents
```

```
Private Const CP_OFF As Byte = 0
```

```
Private Const CP_ON As Byte = 1
```

```
5
```

```
'-----  
' Needs to be executed before the first sending operation  
' (sets initial/stand-by state).  
'-----
```

```
10
```

```
Public Sub CP_InitSend(Path As String, _  
    Link As Integer, _  
    LinkFile As String, _  
    DataFile As String)  
    CP_Abort = False  
    LinkFile = Path + "Cpff" + Trim(Str(Link))  
    Call CP_SendFlag(LinkFile, CP_Source, CP_OFF)  
    DataFile = LinkFile + ".dat"
```

```
15
```

```
End Sub
```

```
20
```

```
'-----  
' Needs to be executed before the first receiving operation  
' (sets initial/stand-by state).  
'-----
```

```
25
```

```
Public Sub CP_InitReceive(Path As String, _  
    Link As Integer, _  
    LinkFile As String, _  
    DataFile As String)  
    CP_Abort = False  
    LinkFile = Path + "Cpff" + Trim(Str(Link))  
    Call CP_SendFlag(LinkFile, CP_Destination, CP_OFF)  
    DataFile = LinkFile + ".dat"
```

```
30
```

```
End Sub
```

```
35
```

```
'-----  
' Handles the notifications after data is written to the Data File.  
'-----
```

```
40
```

```
Public Sub CP_SendData(LinkFile As String)
```

```

    Call CP_SendFlag(LinkFile, CP_Source, CP_ON)
    Call CP_ReceiveFlag(LinkFile, CP_Source, CP_ON)
    Call CP_SendFlag(LinkFile, CP_Source, CP_OFF)
    Call CP_ReceiveFlag(LinkFile, CP_Source, CP_OFF)
5 End Sub

```

```

'-----
' Places in idle state while waiting for data.
'-----

```

```

10 Public Sub CP_WaitForData(LinkFile As String)
    Call CP_ReceiveFlag(LinkFile, CP_Destination, CP_ON)
End Sub

```

```

15 '-----
' Handles the notifications after data is read from the Data File.
'-----

```

```

20 Public Sub CP_ReceiveData(LinkFile As String)
    Call CP_SendFlag(LinkFile, CP_Destination, CP_ON)
    Call CP_ReceiveFlag(LinkFile, CP_Destination, CP_OFF)
    Call CP_SendFlag(LinkFile, CP_Destination, CP_OFF)
End Sub

```

```

25 '-----
' Sends a given flag to a given flag file.
' N.B. Source Roles write only to Source Files.
' Destination Roles write only to Destination Files.
'-----

```

```

30 Private Sub CP_SendFlag(LinkFile As String, _
    Role As String, _
    Flag As Byte)
    Dim Ext As String

35    If CP_Abort = True Then Exit Sub
    If Role = CP_Source Then Ext = "." + CP_Source
    If Role = CP_Destination Then Ext = "." + CP_Destination

```

```

40 OpenAgain:

```



On Error GoTo OpenError

Open LinkFile + Ext For Output Access Write Shared As #1

Write #1, Flag

5 Close #1

Exit Sub

OpenError:

DoEvents

10 If CP\_Abort = True Then Exit Sub

Resume OpenAgain

End Sub

15 '-----  
 ' Waits for a given flag to be in a given flag file.  
 ' N.B. Source Roles read only from Destination Files.  
 ' Destination Roles read only from Source Files.  
 '-----

20 Private Sub CP\_ReceiveFlag(LinkFile As String, \_  
 Role As String, \_  
 Flag As Byte)

Dim Ext As String

Dim Current As Byte

25

If CP\_Abort = True Then Exit Sub

If Role = CP\_Source Then Ext = "." + CP\_Destination

If Role = CP\_Destination Then Ext = "." + CP\_Source

30

Do

If Dir(LinkFile + Ext) < > "" Then

Open LinkFile + Ext For Input Access Read Shared As #1

On Error Resume Next

Input #1, Current

35

On Error GoTo 0

Close #1

End If

DoEvents

Loop While Current < > Flag And CP\_Abort = False

40

End Sub

KMS Implementation

The following is commented source code for implementing the functionality of the present invention. The KMS Series Metrology Tools from Zygo Corporation, Laurel Brook Road Middlefield, CT equipment control package is used, however those skilled in the art recognize that the functionality is not limited to the specific code example illustrated herein. The procedures are complimentary to the Visual Basic example above and share similar names and calling sequences. This equipment control package does not have any communication capabilities except the ability to read and write ASCII text files. However, this capability allows the use of the present communication protocol to interface such a tool to high level automation software for equipment control and data exchange purposes.

```

15  //:-----
    //: 300pc      V3.0
    //:
    //: KMS implementation of Communication Protocol for Data Exchange
    //: via Shared Files.
20  //:
    //: Input variables:
    //:   V40   communication link to be used as data SOURCE
    //:   V41   communication link to be used as data DESTINATION
    //:
25  //: Working variables:
    //:   V00, V01, V02, V03, V04
    //:
    //:-----
    MARK: CP_INIT_SEND
30  MATH: V03 = V40
    MATH: V04 = 0
    CALL: 1 CP_SEND_FLAG
    RETURN:
35
    //:-----
    MARK: CP_INIT_RECEIVE

```

MATH: V03 = V41  
MATH: V04 = 0  
CALL: 1 CP\_SEND\_FLAG  
RETURN:

5

//:-----

MARK: CP\_SEND\_DATA  
MESSAGE: "Sending Data ..."  
10 MATH: V03 = V40  
MATH: V04 = 1  
CALL: 1 CP\_SEND\_FLAG

15 MATH: V03 = V40  
MATH: V04 = 1  
CALL: 1 CP\_RECEIVE\_FLAG

20 MATH: V03 = V40  
MATH: V04 = 0  
CALL: 1 CP\_SEND\_FLAG

25 MATH: V03 = V40  
MATH: V04 = 0  
CALL: 1 CP\_RECEIVE\_FLAG  
MESSAGE: "Done!"

RETURN:

//:-----

30 MARK: CP\_WAIT\_FOR\_DATA  
MESSAGE: "Receiving Data ..."  
MATH: V03 = V41  
MATH: V04 = 1  
CALL: 1 CP\_RECEIVE\_FLAG

RETURN:

//:-----

40 MARK: CP\_RECEIVE\_DATA  
MATH: V03 = V41

- 16 -

MATH: V04 = 1  
CALL: 1 CP\_SEND\_FLAG

5 MATH: V03 = V41  
MATH: V04 = 0  
CALL: 1 CP\_RECEIVE\_FLAG

10 MATH: V03 = V41  
MATH: V04 = 0  
CALL: 1 CP\_SEND\_FLAG  
MESSAGE: "Done!"  
RETURN:

15 //:-----  
//: V03 -- communication link number (SOURCE or DESTINATION)  
//: V04 -- the flag to be written (0 or 1)  
//:  
//: Source -- writes to "\*.0" files  
20 //: Destination -- writes to "\*.1" files  
//:-----  
MARK: CP\_SEND\_FLAG  
MATH: V01 = V03=V41

25 MATRIXOPEN: V00 1 1 "D:\CPFF%.%", V03, V01  
MATRIXSET: V00 1 1 V04  
MATRIXWRITE: V00  
MATRIXCLOSE: V00  
RETURN:

30

35 //:-----  
//: V03 -- communication link number (0-9999)  
//: V04 -- the flag to wait for (0 or 1)  
//:  
//: Source -- reads from "\*.1" files  
//: Destination -- reads from "\*.0" files  
//:-----

40 MARK: CP\_RECEIVE\_FLAG  
MATH: V01 = V03=V40

DOCT "B" 00000000

MATRIXCLOSE: V00

RETURN:

It will be appreciated by those skilled in the art that changes could be made to the embodiments described above without departing from the broad inventive concept thereof. It is understood, therefore, that this invention is not limited to the particular embodiments disclosed, but is intended to cover modifications within the spirit and scope of the present invention.

What is claimed is:

1. A computing system for exchanging data between a first and second computer application of the system, comprising:
  - a computer application data file for receiving data from the first computer application;
  - a computer application send file for receiving notification when the computer application data file has received data from the first computer application;
  - a computer application read file for receiving notification when data has been read from the computer application data file by the second computer application, the first computer application monitoring the computer application read file for notification from the second computer application to initiate further writing to the computer application data file.
2. A computing system for exchanging data between computer applications of the system, comprising:
  - a computer application data file for each computer application for receiving data from a corresponding one of the computer applications;
  - a computer application send file corresponding to each computer application data file for receiving notification when the corresponding computer application data file has received data from the corresponding one of the computer applications;
  - a computer application read file corresponding to each computer application data file for receiving notification when data has been read from the corresponding computer application data file by a non-corresponding computer application, the corresponding computer application monitoring the computer application read file for notification to initiate further writing to the corresponding computer application data file.

1           3.     A method of exchanging data between a first and second  
2 computer application of a computer system, comprising the steps of:

3           writing data of the first computer application to a first computer  
4 application data file;

5           notifying a first computer application send file when data has been  
6 written to the data file by the first computer application;

7           monitoring the first computer application send file from the second  
8 computer application for notification when data has been written to the  
9 first computer application data file by the first computer application;

10          reading the data of the first computer application data file from the  
11 second computer application upon detection of notification;

12          notifying a first computer application read file that data has been read by  
13 the second computer application from the first computer application data  
14 file; and

15          monitoring the first computer application read file from the first  
16 computer application for notification that data has been read from to the  
17 first computer application data file by the second computer application to  
18 initiate further writing to the first computer application data file.

1           4.     The method of exchanging data of claim 3, further comprising the  
2 step of:

3           initializing the contents of the first computer application read and send  
4 files prior to data exchange to enable overwriting of any content therein.

1           5.     The method of exchanging data of claim 3, wherein the computer  
2 system is a network computer system.

1           6.     The method of exchanging data of claim 3, wherein the computer  
2 system is a stand-alone computer system.

1           7.     A method of exchanging bi-directional data between a first and  
2 second computer applications of a computer system, comprising the steps of:

3           writing data of the first computer application to a first computer  
4 application data file;

5           notifying a first computer application send file when data has been  
6 written to the first computer application data file by the first computer  
7 application;

8           monitoring the first computer application send file from the second  
9 computer application for notification that data has been written to the  
10 data file by the first computer application;

11          reading the data of the first computer application data file from the  
12 second computer application upon detection of notification;

13          notifying a first computer application read file when data has been read  
14 by the second computer application from the first computer application  
15 data file;

16          monitoring the first computer application read file from the first  
17 computer application for notification that data has been read from to the  
18 first computer application data file by the second computer application to  
19 initiate further writing to the first computer application data file;

20          writing data of the second computer application to a second computer  
21 application data file;

22          notifying a second computer application send file when data has been  
23 written to the second computer application data file by the second  
24 computer application;

25          monitoring the second computer application send file from the first  
26 computer application for notification that data has been written to the  
27 second computer application data file by the second computer  
28 application;



29 reading the data of the second computer application data file from the  
30 first computer application upon detection of notification;  
  
31 notifying a second computer application read file when data has been  
32 read by the first computer application from the second computer  
33 application data file; and  
  
34 monitoring the second computer application read file from the second  
35 computer application for notification that data has been read from to the  
36 second computer application data file by the first computer application to  
37 initiate further writing to the second computer application data file.

1 8. The method of exchanging data of claim 7, further comprising the  
2 step of:

3 initializing the contents of the shared read and send files prior to data  
4 exchange to enable overwriting of any content therein.

1 9. The method of exchanging data of claim 7, wherein the computer  
2 system is a network computer system.

1 10. The method of exchanging data of claim 7, wherein the computer  
2 system is a stand-alone computer system.

## ABSTRACT OF THE DISCLOSURE

A system and associated method of exchanging data between a first and second computer application of a computer system. Data originating from the first computer application is written to a first computer application data file. A first computer application send file is notified when data has been written to the first computer application data file by the first computer application. The first computer application send file is monitored from the second computer application for notification when data has been written to the first computer application data file by the first computer application. The data of the first computer application data file is read from the second computer application upon detection of notification. A first computer application read file is notified that data has been read by the second computer application from the first computer application data file.

The diagram illustrates a mutual exclusion protocol between two processes, A and B, using semaphores and shared data structures. The protocol is divided into two main sections, A and B, each containing a sequence of operations and shared data structures.

**Process A (Top Section):**

- Operations:**
  - 1. Write (data)
  - 2. Notify (1)
  - 3. Wait for data (1)
  - 4. Read (data)
  - 5. Done (1)
  - 6. Wait for done (1)
  - 7. Clear (0)
  - 8. Wait for clear(0)
  - 9. Clear (0)
  - 10. Wait for clear (0)
- Shared Data Structures:**
  - A\_data.snt**: A semaphore used for mutual exclusion. It is incremented by 1 in step 2 and decremented by 1 in step 7.
  - A\_data.dat**: A shared data structure. It is written to in step 1 and read from in step 4.
  - A\_data.red**: A shared data structure. It is incremented by 1 in step 5 and decremented by 1 in step 9.

**Process B (Bottom Section):**

- Operations:**
  - 1. Write (data)
  - 2. Notify (1)
  - 3. Wait for data (1)
  - 4. Read (data)
  - 5. Done(1)
  - 6. Wait for done (1)
  - 7. Clear (0)
  - 8. Wait for clear(0)
  - 9. Clear (0)
  - 10. Wait for clear (0)
- Shared Data Structures:**
  - B\_data.snt**: A semaphore used for mutual exclusion. It is incremented by 1 in step 2 and decremented by 1 in step 7.
  - B\_data.dat**: A shared data structure. It is written to in step 1 and read from in step 4.
  - B\_data.red**: A shared data structure. It is incremented by 1 in step 5 and decremented by 1 in step 9.

The diagram shows the flow of operations and the state of shared data structures for both processes. The operations are numbered 1 through 10, and the shared data structures are labeled A\_data.snt, A\_data.dat, A\_data.red, B\_data.snt, B\_data.dat, and B\_data.red. The operations are connected by arrows indicating the sequence of execution. The shared data structures are represented by boxes, and the operations are represented by text labels.

FIG. 1

## Declaration and Power of Attorney

My residence, post office address and citizenship are as stated below next to my name.

I hereby state that I have reviewed and understand the contents of the above identified specification, including the claims, as amended by an amendment, if any, specifically referred to in this oath or declaration.

I hereby claim foreign priority benefits under Title 35, United States Code, 119 of any foreign application(s) for patent or inventor's certificate listed below and have also identified below any foreign application for patent or inventor's certificate having a filing date before that of the application on which priority is claimed:

I hereby claim the benefit under Title 35, United States Code, 120 of any United States application(s) listed below and, insofar as the subject matter of each of the claims of this application is not disclosed in the prior United States application in the manner provided by the first paragraph of Title 35, United States Code, 112, I acknowledge the duty to disclose all information known to me to be material to patentability as defined in Title 37, Code of Federal Regulations, 1.56 which became available between the filing date of the prior application and the national or PCT international filing date of this application:

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

Figure 1 consists of 10 sub-graphs labeled (a) through (j), each showing the growth of *P. aeruginosa* over a 24-hour period. The y-axis for all graphs is 'Growth (OD600)' ranging from 0.0 to 1.0, and the x-axis is 'Time (h)' ranging from 0 to 24. The growth curves are as follows:

- (a) Control: Shows a typical growth curve, reaching an OD600 of approximately 0.8 by 24 hours.
- (b) 100 µg/ml: Shows a growth curve similar to the control, reaching an OD600 of approximately 0.8 by 24 hours.
- (c) 200 µg/ml: Shows a growth curve similar to the control, reaching an OD600 of approximately 0.8 by 24 hours.
- (d) 400 µg/ml: Shows a growth curve similar to the control, reaching an OD600 of approximately 0.8 by 24 hours.
- (e) 800 µg/ml: Shows a growth curve similar to the control, reaching an OD600 of approximately 0.8 by 24 hours.
- (f) 1600 µg/ml: Shows a growth curve similar to the control, reaching an OD600 of approximately 0.8 by 24 hours.
- (g) 3200 µg/ml: Shows a growth curve similar to the control, reaching an OD600 of approximately 0.8 by 24 hours.
- (h) 6400 µg/ml: Shows a growth curve similar to the control, reaching an OD600 of approximately 0.8 by 24 hours.
- (i) 12800 µg/ml: Shows a growth curve similar to the control, reaching an OD600 of approximately 0.8 by 24 hours.
- (j) 25600 µg/ml: Shows a growth curve similar to the control, reaching an OD600 of approximately 0.8 by 24 hours.

04/00

David Volejnicek	(Reg. No. 29355)
Charles L. Warren	(Reg. No. 27407)
Jeffrey M. Weinick	(Reg. No. 36304)
Eli Weiss	(Reg. No. 17765)

I hereby appoint the attorney(s) on ATTACHMENT A as associate attorney(s) in the aforementioned application, with full power solely to prosecute said application, to make alterations and amendments therein, to receive the patent, and to transact all business in the Patent and Trademark Office connected with the prosecution of said application. No other powers are granted to such associate attorney(s) and such associate attorney(s) are specifically denied any power of substitution or revocation.

Full name of sole inventor (or 1st joint inventor): **Valentin Panayotov**

Inventor's signature Valentin Panayotov Date 10/30/2000

Residence: Sinking Spring, Pennsylvania

Citizenship: Bulgaria

Post Office Address: 14 Illinois Avenue

Sinking Spring, Pennsylvania 19608

ATTACHMENT A

Paul F. Prestia	Reg. No. 23,031
Allan Ratner	Reg. No. 19,717
Andrew L. Ney	Reg. No. 20,300
Kenneth N. Nigon	Reg. No. 31,549
Kevin R. Casey	Reg. No. 32,117
Benjamin E. Leace	Reg. No. 33,412
James C. Simmons	Reg. No. 24,842
Lawrence E. Ashery	Reg. No. 34,515
Christopher R. Lewis	Reg. No. 36,201
Robert L. Andersen	Reg. No. 25,771
Joshua L. Cohen	Reg. No. 38,040
Daniel N. Calder	Reg. No. 27,424
Louis W. Beardell, Jr.	Reg. No. 40,506
Jacques L. Etkowicz	Reg. No. 41,738
Jack J. Jankovitz	Reg. No. 42,690
Jonathan H. Spadt	Reg. No. 45,122
Kevin W. Goldstein	Reg. No. 34,608
Christopher I. Halliday	Reg. No. 42,621
Scott A. Mckeown	Reg. No. 42,866

Telephone calls should be made to Allan Ratner of Ratner & Prestia at:

Phone No.: (610) 407-0700

Fax No.: (610) 407-0701

All written communications are to be addressed to:

Allan Ratner  
Ratner & Prestia  
Suite 301, One Westlakes (Berwyn)  
P.O. Box 980  
Valley Forge, PA 19482-0980